Appendix F

Title:
Macro Specification DMUR Version 2.2

1 Introduction

1.1 Overview

The Receive Data Management Unit (DMUR) provides direct data transfer from the on-chip Receive Buffer (RB) to the shared memory with minimal host CPU intervention. The on-chip Receive Buffer stores data after protocol (HDLC, Ethernet, ATM,...) processing for each channel, so that the DMUR can initiate a data burst on the system bus (e.g. PCI) and decrease the bus occupancy. A linked list of descriptors associated to each channel is located in the shared RAM and handled by the DMUR. The address generator of the DMUR supports full link list handling. The descriptors can be stored in separate memory location from the data buffers themselves allowing full scatter/gather assembly of packets. In order to have only one read access on the system bus for each descriptor, the current descriptor is hold on-chip. The interrupt vectors generated by the DMUR are transferred to a separate interrupt controller (DMUI). The DMUI will transfer the interrupt vectors of the complete device to the shared memory.

The number of channels is a flexible parameter of the DMUR macro. The DMUR can be used for multichannel devices (M256) as well as for high speed controllers with a reduced number of channels (DSCC4).

1.2 Features

- Data transfer from the central Receive Buffer (RB) to the shared memory
- Support of linked list data structures located in the shared memory and consisting of a descriptor associated to a data section
- Independent channel configuration
- Forward interrupt vectors to the DMUI

1.3 System Integration

The DMUR has four interfaces:

- FPI Initiator Bus
- FPI Target Bus
- RB interface
- DMUI interface

The protocol data is read at the RB interface and written to the FPI initiator bus. The descriptors are accessed through the FPI initiator bus. The interrupt vectors are transferred on the DMUI interface bus. Each channel is configured (base address register, command register,...) via the FPI target bus.

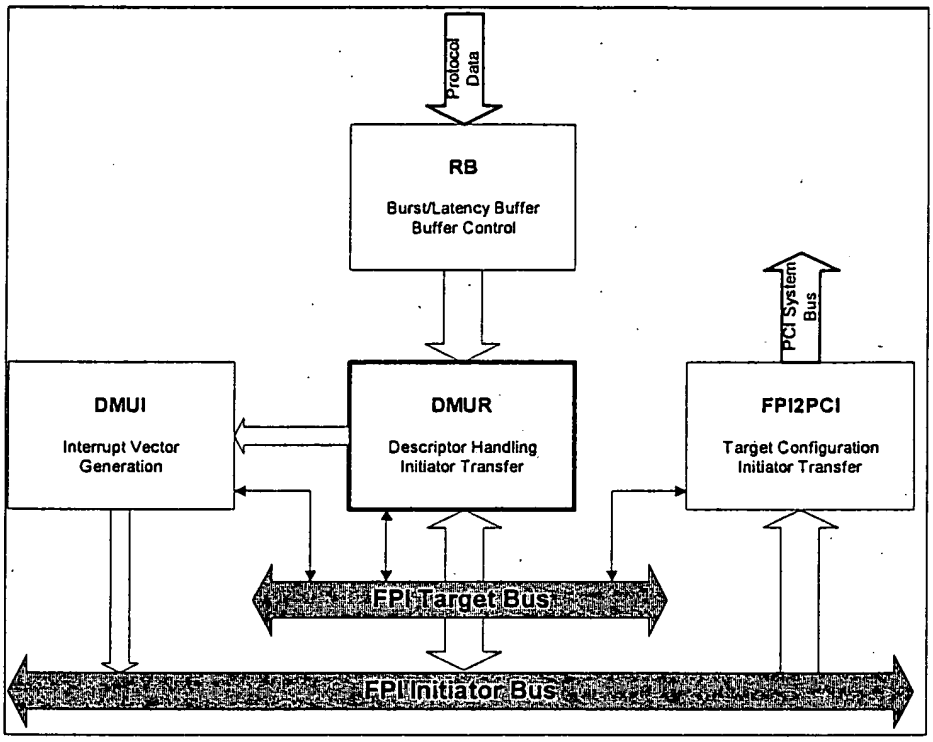


Figure 1
System Integration

1.4 Known Restrictions and Problems

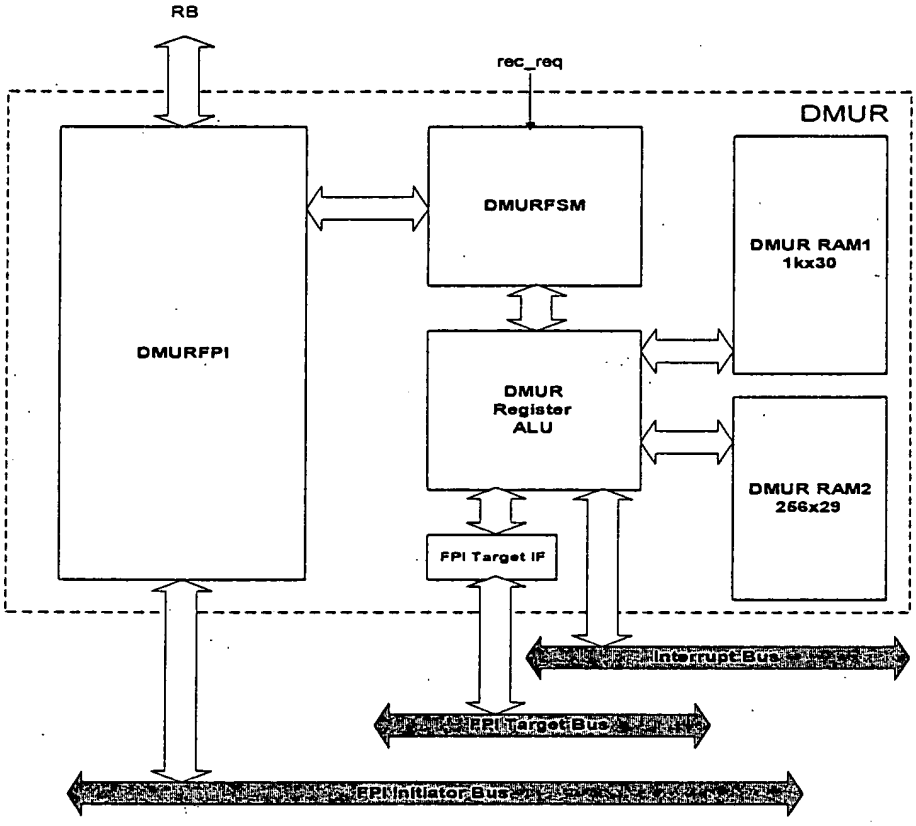
The priorization on the FPI Initiator bus is not an easy task.

DMUR should have priority if new data or a descriptor change is required in the middle of a frame in order to prevent an overflow of the RB (and an abort of the frame). But at the beginning of a frame, the transfer is not time critical.

DMUR and DMUI should have an equal priority on the FPI initiator bus to prevent buffer overflow.

2 Functional Description

2.1 Block Diagram incl. Clocking Regions



The DMUR includes 2 RAMs that store the current descriptor, the next descriptor, the amount of the already transferred data, the amount of left double words in the data section, the status, the mode, the interrupt queue number and the interrupt mask for each channel. The controller unit (DMURFSM) initiates and controlls the data transfer from the receive buffer to the FPI Initiator Bus, the read and the update of the receive descriptors. In order to achieve a maximum data throughput the DMURFPI is directly connected to the receive buffer and the Initiator bus and transfers the data FPI bus compliant, triggerd by the DMURFSM. The ALU and the register set are necessary for burst length and address calculations.

The complete DMUR block and all the interfaces are synchronous to the same CLK signal derived from the FPI initiator bus.

2.2 Normal Operation Description

The DMUR operates with a linked list of descriptors pointing to a data section. For each active channel a linked list is allocated in the external shared memory by the CPU. The descriptor contains the pointer to the next descriptor, the start address of the data section and the reserved size of the data section. The DMUR will update the descriptor with an additional field consisting of the effective number of bytes written in the data section.

For initialization of a channel in the DMUR, the CPU programs the First Receive Descriptor Address (FRDA) register, the interrupt queue, the interrupt mask and the Channel specification Command register with the Receive Init Command (refer to chapter 4.2). These information will be stored at the internal RAM. When the receive buffer requests a data transfer for this channel the DMUR then fetches the descriptor pointed by FRDA from the shared memory, which in turn point to the data buffer and its length as well as the next receive descriptor to go after the reception of the current buffer is completed.

The DMUR gets requests from the Receive Buffer for the transfer of data. The DMUR reads first the request register (located at address 0 on the bus between DMUR and RB). RB returns in the following clock cycle the Request Register, consisting of the channel number (CHN) and the number of words (BL+1) the RB wants to transfer to DMUR and a flag whether the last word contains status information (Data/Status Flag = 1) or not (Data/Status Flag = 0). If the last word in the transfer is a status word, the number of valid bytes in this status word will be provided additionally using the BE bits. In this case the BE bits have the following meaning:

Table 1
Request Register

Bit	31... 27	26	25	24	23.. BLB+16	BLB+15...16	15 .. CNB	CNB-1 0
Function	Reserved	BE 1	BE 0	Data/ Status Flag	Reserved	Burst length BL	Reserved	Channel number (CHN)

BE[0..1]=00: Byte 0-2 are invalid, the word contains only status information

BE[0..1]=10: Byte 0 is valid, Byte 1-2 are invalid user data.

BE[0..1]=01: Byte 0-1 are valid, Byte 2 is invalid user data.

BE[0..1]=11: Byte 0-2 are valid

The DMUR reads the Receive Descriptor corresponding to the channel number CHN and calculates the maximum number of bytes that can be transferred in the current receive data section in the external memory by checking the Byte Number (NO) field.

If NO/4 is greater or equal than the Burst Length (BL+1) value, the DMUR will start a burst transfer with BL+1 words from the Receive Buffer to the FPI2PCI bridge over the

FPI initiator bus. Otherwise, if NO/4 is less or equal than (BL+1), the DMUR will start a Burst Transfer with NO/4 words.

The DMURFSM calculates whether (BL+1) or NO/4 words are transferred and requests the transfer from the DMURFPI. Additionally the start address for the data in the shared memory will be a parameter for the DMURFPI. The DMURFPI reads the protocol data on the RB interface at the data port register (address 1, first DWORD) and stores it internally. The DMUR will then arbitrate the FPI master bus. Once it was granted it will transfer the stored DWORD to the bridge and continue transfering the rest of the burst from the receive buffer directly to the FPI2PCI bridge.

If the data section in the shared memory has been filled with data, the DMUR will update the receive descriptor by writing the complete (C) bit and the number of stored bytes (BNO) in the current data section. An HI interrupt vector will be transferred to the DMUI if the HI bit is set in the receive descriptor. The DMUR will then branch to the next receive descriptor.

In the case where the data section is not full and the frame is not yet ended (with a valid or invalid status), the receive descriptor is not updated in the shared memory (only the on-chip BNO value is updated).

This procedure will continue until the complete Burst Length (available data) is transferred to the shared memory through the FPI initiator bus.

When a frame end is detected and all data have been transferred, the DMUR writes the number of bytes (BNO) stored in the data section in the current descriptor, sets the Frame End (FE) bit and writes the complete (C) bit, no matter if the datasection is full or not. The status of the completed frame will be additionally written into the Receive Descriptor. A FE interrupt vector is transferred to the DMUI, the HI will be set in this vector as well if the corresponding bit is set in the current receive descriptor.

The data transfer is controlled by the HOLD bit, which is located in the Receive Descriptor. If the HOLD bit is not set the current receive descriptor is not the last one in the descriptor chain for this channel and the DMUR will then branch off to the next descriptor.

If the current receive descriptor has its "HOLD" bit set, it is the last one in the descriptor chain. Four different cases have to be distinguished:

a) When the current data buffer has been filled, does not contain the end of a frame (frame based protocolls) and the requested burst length by the RB is not satisfied, the DMUR polls the HOLD bit of the current Receive Descriptor once more. If then the hold bit is removed, the DMUR branches to the next descriptor (NRDP). If the HOLD Bit is still set, an internal Poll Bit will be set and the DMUR does not branch off to the next descriptor for this channel. Additionally an Hold Caused Receive Abort Interrupt (HRAB, HI bit set if necessary) is generated. The status of the descriptor in the shared memory is aborted (RAB bit set), the Complete and the Frame End Bit will be set in the Receive Descriptor. The rest of the frame will be discarded by the DMUR. As long as the HOLD bit remains set, the transferred data from RB for this channel are discarded by the DMUR

and for each discarded frame an interrupt with the bits HRAB and RAB set will be generated. No descriptor update will take place.

b) The current data buffer has been filled up and it does contain the end of frame. A Frame End Interrupt (HI bit set if necessary) will be generated, the descriptor will be updated with the FE and C bit and the status of this receive descriptor is error free. With the next request for this channel by the receive buffer, the DMUR repolls the HOLD bit of the current receive descriptor. If then the hold bit is removed, the DMUR branches to the next descriptor (NRDP). If the HOLD Bit is still set, an internal Poll Bit will be set and the DMUR does not branch off to the next descriptor for this channel. As long as the HOLD bit remains set, the transferred data from RB for this channel are discarded by the DMUR and for each discarded frame an interrupt with the bits HRAB and RAB set will be generated. No descriptor update will take place.

c) The current data buffer has been filled up completely (requested transfer length = current NO) and does not contain the end of the frame. The descriptor will be updated immediately (C bit set), an HI Intr will be generated if the HI bit is set. With the next request for this channel by the receive buffer, the DMUR repolls the HOLD bit of the current receive descriptor. If then the hold bit is removed, the DMUR branches to the next descriptor (NRDP). If the HOLD Bit is still set, an internal Poll Bit will be set and the DMUR does not branch off to the next descriptor for this channel. Additionally an Hold Caused Receive Abort Interrupt (HRAB, HI bit set if necessary) is generated. No descriptor update will take place and the data of the current receive buffer request will be discarded. As long as the HOLD bit remains set, the transferred data from RB for this channel are discarded by the DMUR and for each discarded frame an interrupt with the bits HRAB and RAB set will be generated. No descriptor update will take place.

d) The current receive descriptor has its HOLD bit set, but the requested transfer length by the receive buffer is smaller than the reserved data section (NO). All data will be transferred to the reserved data section, no descriptor update will take place neither an interrupt will be generated. This procedure will continue until either case a), case b) or case c) will become true.

When the next receive descriptor is available in the shared memory, the CPU can write the Channel Specification command register (refer to chapter chapter 4.2) with the Receive Hold Reset (RHR) Command, so the internally Poll Bit will be cleared. When the RB requests a new data transfer for this channel, the DMUR still sees the onchip stored HOLD Bit set, but the Poll Bit reseted.This combination forces the DMUR to repoll the HOLD descriptor. The HOLD bit is supposed to be removed, additionally the Next Receive Descriptor Pointer will be read. Then it will branch to this Next Receive Descriptor Pointer. After the RHR command,

- for a frame based protocoll (refer to chapter chapter 4.2.4) the data transferred from RB is discarded until the end of a received frame and the next received frame is related to the next receive descriptor.

– for the transparent mode the data transferred from RB is written immediately to the data section of the next receive descriptor.

If the CPU issues a Receive Hold Reset Command and does not remove the HOLD bit (erroneous programming), no action will take place. Alternatively a Receive Abort or a Receive Off command can be issued by the CPU in order to release a channel from this HOLD state (refer to command description).

If the big endian byte ordering is programmed, the DMUR provides byte ordering swapping for the data transfer between RB and the FPI initiator bus.

Note: The descriptor transfer is always a 32 bit access (no byte swapping).

The following table shows how data is stored in the shared memory depending on the little or big endian byte ordering convention:

Table 2
Little/Big Endian Byte Ordering

BNO	Little Endian				Big Endian			
3	-	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	-
7	Byte3	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	Byte3
	-	Byte6	Byte5	Byte 4	Byte 4	Byte5	Byte6	-

2.3 Receive Descriptor Description

The receive descriptor is initialized by the host CPU in the shared memory and is read by the DMUR, when requested so by the host via Receive Init command or after branching from the previous receive descriptor. The receive descriptor is read entirely at the beginning and stored in the on-chip memory. Therefore all information in the next descriptor must be valid when DMUR branches to this descriptor.

FE, C, BNO and status are set by the DMUR, all other values are set by the host.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Byte Address																
0x00	Rsvd	HOLD	HI	Offset			Rsvd				Descriptor ID					
0x04	Next Receive Descriptor Pointer															
0x08	Receive Data Pointer															
0x0C	FE	C	Rsvd								Status					

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte Address																
0x00	NO															
0x04	Next Receive Descriptor Pointer															
0x08	Receive Data Pointer															
0x0C	BNO															

HOLD: Hold

It indicates whether the current descriptor is the last element of a linked list or not:

HOLD=0: A next descriptor is available in the shared memory; after checking the HOLD bit stored in the on-chip memory the DMUR branches to next receive descriptor.

HOLD=1: The current descriptor is the last one for this channel that is available for the DMUR. After completion of the current receive descriptor an HRAB interrupt is generated if the complete frame could not be stored in the data section. The corresponding DMUR channel is deactivated for receive direction as long as the CPU does not request an activation via the Channel Specification command register.

HI: Host Initiated Interrupt

If the HI bit is set, the DMUR generates an HI interrupt vector to the interrupt controller (e. g. DMUI) after transferring all relevant data bytes into the current data section.

NO: Byte Number

This byte number defines the size of the receive data section allocated by the host. The maximum buffer length is 65535 bytes and it has to be a multiple of 4 bytes. The data bytes are stored into the receive data section according to the selected mode (little endian or big endian).

If NO is set to 0, no data will be written to the datasection. The DMUR updates the status of the current receive descriptor with BNO = 0 and sets the Complete Bit. An HI intr will generated, if necessary. Afterwards the DMUR jumps immediately to the next receive descriptor.

Next Receive Descriptor Pointer:

This 32-bit pointer contains the start address of the next receive descriptor. After completion of the current receive descriptor the DMUR branches to the next receive descriptor to continue reception. The receive descriptor is read entirely at the beginning of reception and stored in on-chip memory. Therefore all information in the next descriptor must be valid when the DMUR branches to this descriptor.

This pointer is not used if a receive abort command is detected while the DMUR still writes data to the current receive descriptor. In this case the First Receive Descriptor Address (FRDA) is used as a pointer for the next receive descriptor to be branched to.

Receive Data Pointer:

This 32-bit pointer contains the start address of the receive data section. The start address must be 32-bit (DWORD) aligned.

FE: Frame End

It indicates that the current receive data section (addressed by Receive Data Pointer) contains the end of a frame (frame can be valid or invalid - further information in the status). This bit is set by the DMUR after transferring the last data of the frame from the internal receive buffer into the receive data section in the shared memory. Moreover the BNO is updated, the C bit is set by the DMUR and a Frame End Interrupt is generated.

With the next request for this channel, the DMUR checks the HOLD bit stored in on-chip memory. When HOLD=0 it branches to the next receive descriptor. Otherwise the corresponding DMUR channel is deactivated as long as the CPU does not request an activation via the command register.

C: Complete

This bit is set by the DMUR if

- it completes filling data section normally
- it was aborted by a receiver reset command
- the data sections contains a end of frame (for frame based protocols)

was stored in the receive data section.

BNO: Byte Number of Received Data

The DMUR writes the number of data bytes it has stored in the current data section into BNO field.

Descriptor ID

This field is read by the DMUR and written back in the corresponding interrupt status vector for all channel interrupt vectors. This value provides a link between the interrupt vector and the descriptor, to be used by the software.

Offset

Word (32 bit) offset of unused data section. This field is set by the CPU and is only used by DMUR at the beginning of a frame (will be ignored in transparent mode) and allows the CPU to reserve memory space for an additional header. If the descriptor is the first of a frame the DMUR will write data at the address Receive Data Pointer + Offset, otherwise the field is ignored. The Offset has to be equal or bigger than NO/4. If the Offset is equal to NO/4, the DMUR does not write any data to this datasection, just the complete bit will be set and data will be stored in the data section belonging to the next receive descriptor.

Status

The DMUR writes the status information into the Status byte whenever it sets the C and FE bit field. It is the copy of the status bytes coming from the Receive buffer (refer to chapter 3.2) and has to following coding:

bit 16: RAB

Bit 17: ILEN

Bit 18: CRC

Bit 19: RFOD

Bit 20: MFL

2.4 Interrupts

The DMUR generates 2 kind of interrupts, the command interrupts and interrupts which are related to the transfer mechanism of the DMUR controller for a particualr channel. In order to distinguish these 2 interrupt groups, command interrupts and channel interrupts will be introduced. All command interrupt will be writen to a dedicated queue with a fixed length, channel interrupts can be written to 8 different queues, the designated queue for each channel will assigned during initialization. The interrupt vectors basicly consist of the ID, the channel number and specifific bits.

The Descriptor ID value is written back in the corresponding channel interrupt vector. This provides a link between the interrupt vector and the descriptor, to be used by the software. Additionally the interrupt queue number for this channel will be provided in channel interrupt vectors. The interrupt vectors are transferred to the DMUI and have the following format:

Table 3
Interrupt Vector Format (channel interrupt vector)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	0	1	0	1	Q2	Q1	Q0	0	0	Descriptor ID					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HI	RAB	FE	HRAB	MFL	RFOD	CRC	ILEN	Channel No							

Command Interrupt Vector

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CMDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Channel No							

Channel number

Identifies the channel where the interrupt occurred

Descriptor ID

Identifies the receive descriptor, which generated the interrupt

Q2,Q1,Q0: Queuenumber

Identifies the number of the interrupt queue, located in the shared memory

HI: Host Initiated Interrupt

If the HI bit in the receive descriptor is set, this bit will be activated in the interrupt vector when DMUR finishes the current Receive Descriptor (no matter whether a next descriptor is available or not).

FE: Frame Indication Interrupt

FE=1 indicates, that a frame (valid or invalid) has been received completely or was stopped by a Receive Abort, Receive Off command or a HOLD in a receive descriptor. If the frame is valid or not depends on the status bits RAB, HRAB, MFL, RFOD, CRC, ILEN in the vector. As soon as one of these bits is set, errors occured during the reception of the frame. It is also set when the descriptor in which the frame has been finished contained a hold bit (combination Frame End + HOLD).

HRAB: Hold Caused Receive Abort

Issued if the requested amount of data could not be transferred to the shared memory completely because of a HOLD bit set in the current receive descriptor not providing enough memory space for the requested data. An interrupt with RAB and HRAB set will be generated for every complete frame which was lost due to the HOLD bit in the receive descriptor.

RAB

Receive Abort indication. This bit will be set, when the CPU programmed a receive abort or a receive off command and therefore a frame could not be transmitted completely (abort from the CPU side). Additionally it can be set in the status word coming from the receive buffer indicating that the frame was aborted from the serial side. It then will be copied to the interrupt vector as well. It will be also set in combination with HRAB bit for each discarded frame (refer to HRAB description).

The following 4 bits are copied by the DMUR from the status byte coming from RB in the interrupt vector. It includes status information generated by the Protocol Handler.

MFL

Maximum Frame Length. The Packet exceeds the max. allowed frame size.

RFOD

Indicates a Receive Frame overflow. The Protocol handler was unable to transfer data to the RB. As soon RB can store data again, a RFOD status is appended to the frame.

CRC

Cyclic Redundancy Check error

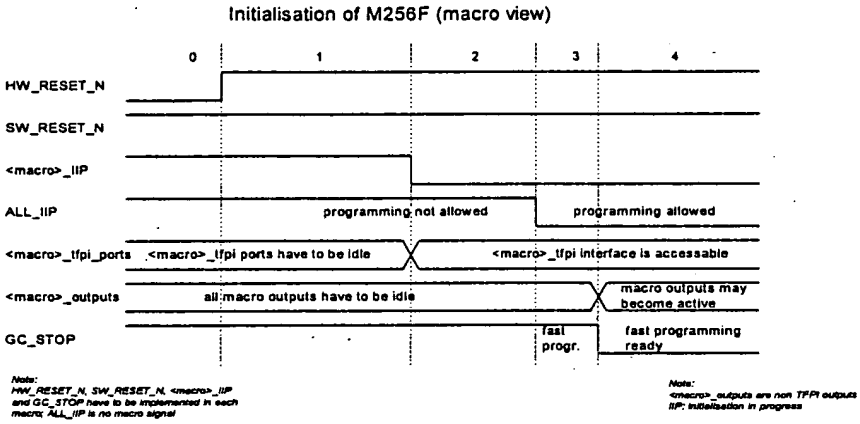
Invalid length (ILEN)

Indicates a frame length not multiple of 8 bits.

CMDC: Command Completed

Identifies a successful channel command performed by the host CPU. The DMUR can accept a new command for this channel.

2.5 Reset Behavior



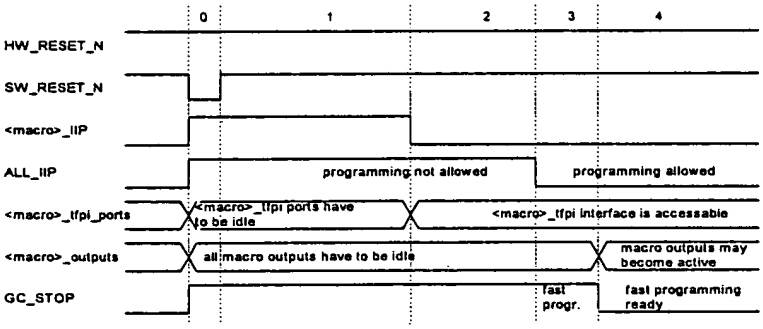
0: HW_RESET_N is active and deactivates SW_RESET_N; all macros are reset; GC_STOP set to '1'

1: HW_RESET_N inactive => each macro executes the macro specific initialisation of rams

2: some macros are ready with initialisation of rams, some not (ALL_IIP is a or-function of all <macro>_IIPs). The "ready macros" could be accessed by TFPI interface, but software programming is not allowed as long as ALL_IIP is active

3: all macros are ready with initialisation => ALL_IIP becomes inactive; software polls the ALL_IIP bit after HW_RESET_N; if ALL_IIP = 0 then software programms all macros (in a default "fast programming mode", GC_STOP = 1; reset value of GC_STOP is '1', but can be reset at any time)

4: after fast programming macro outputs (non TFPI) may become active and macro has to react on non FPI inputs



0: SW_RESET_N becomes active; all macros (registers) are reset; GC_STOP is active

1: SW_RESET_N inactive => each macro executes the macro specific initialisation of rams

2: some macros are ready with initialisation of rams, some not (ALL_IIP is a or-function of all <macro>_IIPs). The "ready macros" could be accessed by TFPI interface, but software programming is not allowed as long as ALL_IIP is active

3: all macros are ready with initialisation => ALL_IIP becomes inactive; software polls the ALL_IIP bit after SW_RESET_N; if ALL_IIP = 0 then software programms all macros in a "fast programming mode", GC_STOP = 1

4: after fast programming macro outputs (non TFPI) may become active and each macro has to react on non FPI inputs

All internal registers and functions are initialized to known states (RESET state).

In addition to this global reset, each DMA channel can be reseted (turned off) via the command register.

2.6 Functional Test Description

2.7 Production Test Description

3 Macro Interfaces and Signal Description

All signals are active high until otherwise specified. Active low signals are designated by "_N" (FPI mode) appended to their names. To make the design as re-usable as possible, a bus signal whose width is application dependent is specified with one of the following parameters:

Parameter name	Bus Type	Typical value (bits)
		M256F
CNB	Channel Number Bus	8
DBB	Data/Status Bus	32
BLB	Burst Length Bus	6

Note: Since the maximum burst length is limitied to 64 DWORDS by the PCI2FPI bridge in the M256F the Receive Buffer will never request a data transfer exceeding this limit

3.1 Signal Description

In the following sections, "Flexible Peripheral Interconnect (FPI) Bus compliant" means that the specified bus uses a subset of the FPI features and satisfies the basic address and data cycle. Not all FPI signals are implemented because default values are sufficient for the application i.e. they can be coded as constants in the hardware. Refer to the FPI bus specification for details of the complete bus.

Two additional out-band signals were introduced at the receive buffer - DMUR interface:

- REC_STAT to indicate if currently transferred word is status instead of pure data. This signal is valid for one cycles in DMA transfer.
- REC_REQ_N to indicate that the Receive Buffer Action Queue (RBAQ) in the RB is not empty.

Table 4
Macro Interfaces and Signal Description

Symbol name	I/O	Function
Control Signals		
sysclk	I	system clock
reset_n	I	Hardware reset
scanmode	I	Scanmode
gc_stop	I	global stop signal, used for fast RAM programming by the CPU
dr_iip	O	Initialisation in progress, when active the DMUR initialize its internal RAMs
Receive Buffer (RB) interface		
dr_rec_a	O	1 bit address bus 0 => status port (request register). 1 => data port.
dr_rec_rd_n	O	Read Control Output
rec_d[DBB-1:0]	I	Output data/status from RB to DMUR controller
rec_rdy	I	End of data transfer indication. 0 => DMUR should insert wait state. 1 => RB will finish transfer during this clock cycle.
rec_stat	I	Current transferred word is status
rec_req_n	I	Service request from RB to DMUR controller. Asserted when RBAQ is not empty.
Interrupt Controller (DMUI) Interface		
dr_int_req_n	O	Interrupt Bus Request Line
int_gnt_n	I	Interrupt Grant Line (from interrupt bus arbiter)
dr_int_d[DBB-1:0]	O	Interrupt Vector

Table 4
Macro Interfaces and Signal Description (cont'd)

Symbol name	I/O	Function
dr_int_d_en[7:0]	O	Output Enable Bus

FPI Target Interface

tfpi_a[8:2]	I	Address bus.
tfpi_d[DBB-1:0]	I	Data bus input
tfpi_rdy	I	Ready Input, other target will finish data cyle in the clock cycle (tfpi_rdy = 1) or it will insert Waitstates
tfpi_wr_n tfpi_rd_n	I I	Read/Write controls. Following codes are defined: WR_N = 0; RD_N = 1 => data written to DMUR WR_N = 1; RD_N = 0 => data read from DMUR
tfpi_vc_sel_n	I	DMUR Slave select (registers of virtuell channel specification). 0 => Address is valid 1 => Address is not valid
tfpi_vg_sel_n	I	DMUR Slave select (global registers). 0 => Address is valid 1 => Address is not valid
dr_tfpi_d[DBB-1:0]	O	Data bus output
dr_tfpi_d_en[7:0]	O	Output Enable Bus for data bus (one enable line for 4 data lines)
dr_tfpi_rdy	O	End of transfer indicator: 0 => Master should insert wait states 1 => DMUR will complete transfer in this cycle
dr_tfpi_rdy_en	O	Output Enable signal for Ready Output

FPI Initiator Bus

ifpi_gnt_n	I	DMUR INITIATOR Grant Line
ifpi_opc[3:0]	I	Operation Code (Input for split read response)
ifpi_d[DBB-1:0]	I	Data Bus (for split block read)
ifpi_sel_n	I	Select Signal for DMUR (for split block response)
ifpi_ack[1:0]	I	Slave Response code

Table 4
Macro Interfaces and Signal Description (cont'd)

Symbol name	I/O	Function
ifpi_rdy	I	Slave on data bus will be able to finish transaction in current clock cycle
dr_ifpi_req_n	O	DMUR INITIATOR Bus Request Line
dr_ifpi_rd_n	O	Read Control
dr_ifpi_rd_n_en	O	Read Control Output Enable
dr_ifpi_wr_n	O	Write Control
dr_ifpi_wr_n_en	O	Write Control Output Enable
dr_ifpi_abort_n	O	Abort Signal
dr_ifpi_abort_n_en	O	Abort Output Enable
dr_ifpi_opc[3:0]	O	Operation Code
dr_ifpi_opc_en	O	Operation Code Output Enable
dr_ifpi_tc[BLB-1:0]	O	Transfer Count (max. burst size 64dwords)
dr_ifpi_tc_en[1:0]	O	Transfer Count Output Enable (one enable line drives 4 data lines)
dr_ifpi_a[DBB-1:2]	O	Address Bus
dr_ifpi_a_en[7:0]	O	Address Bus Output Enable Bus
dr_ifpi_d[DBB-1:0]	O	Data Bus
dr_ifpi_d_en[7:0]	O	Data Bus Output Enable Bus
dr_ifpi_tag[3:0]	O	Tag Bus (used to transfer the master ID for Split Block Transfers)
dr_ifpi_tag_en	O	Tag Bus Output Enable
dr_ifpi_rdy	O	DMUR will be able to finish transaction in current clock cycle
dr_ifpi_rdy_en	O	Ready Output Enable

(*) These signals connect DMUI and the arbitration logic on the DMUI bus.

3.2 Data Flow and Functional Timing

3.2.1 FPI Initiator Bus

Refer to the SIEMENS FPI Bus for Munich-Macros Specification.

3.2.2 FPI Target Bus

Refer to the SIEMENS FPI Bus for Munich-Macros Specification.

3.2.3 RB Interface

As soon as the receive buffer wants to transfer data to the system memory through the DMUR it asserts its rec_req_n signal. If the DMUR is currently in idle state, it initiates an address cycle by asserting the read signal with address 0. During the data cycle the RB returns the RB-DMUR request register and asserts rec_rdy_n (one wait state will be typically inserted). Based on the requested burst and the channel number, the DMUR set up the transfer by assigning the destination address and the burst length. By asserting the read signal and the address (address 1 data buffer) the DMUR transfers BL+1 words with either individual or overlapped cycles. In best case, a burst cycle can occur without wait states but internally, the PMR and configuration interfaces have higher priority.

Figure 2 illustrates a basic transfer (the RB inserts typically 1 wait state at the beginning of the data transfer, 2 wait states in cycles 10 and 11 are inserted due to collision with the protocoll machine interface). Note that the DMUR needs some cycles to initiate the transfer after reading the request register for loading the channel context and address calculation.

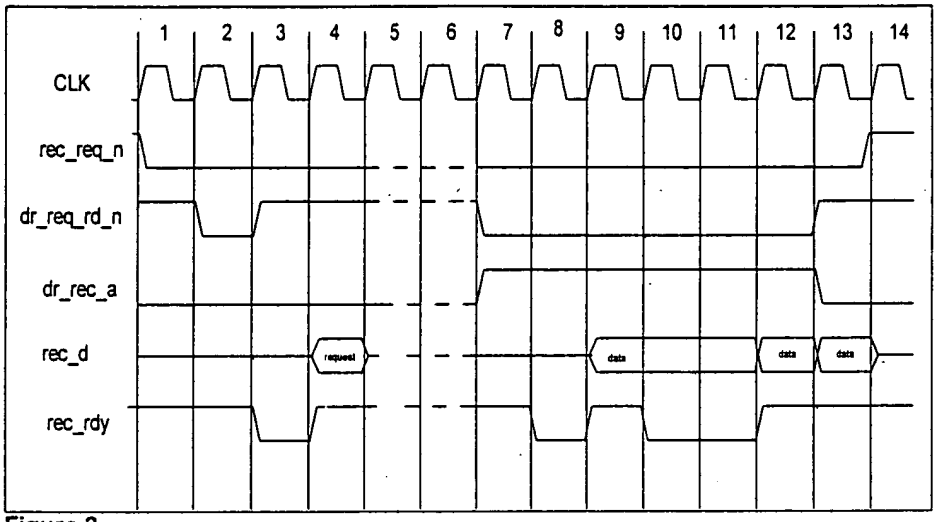


Figure 2
Basic transfer between DMUR and RB

The efficiency of the transfers improves with burst length. The peak throughput (not sustainable) can approach 1 word/clock with long bursts. However it is ultimately limited by PMR. Each word the PMR transfers during a RB-DMUR data cycle insert 2 wait states in RB_RDY_N.

Note that DMUR must always read the request register before a data transfer. The Burst Length field in RBAQ is actualized after each data access and the entry in RBAQ is removed after all the corresponding data have been transferred to the DMUR.

If the burst length ((BL+1) *4) in the request register of the RB is bigger than NO of the current datasection, the DMUR will split the transfer. It will first fill the current datasection, update the status of the current receive descriptor, generate an HI interrupt (if necessary) and branch to the Next Receive Register pointed by the Next Receive Descriptor Pointer. Then it will continue transferring data.

The side band signal rec_stat will be asserted when the current transferred data word at the interface RB-DMUR contains status information (i.e. Frame End).

Figure 3 shows a data transfer to 2 data sections (2 different receive descriptors) and the use of the sideband signal rec_stat (the typical wait state at the beginning of a transfer is not shown in the figure)

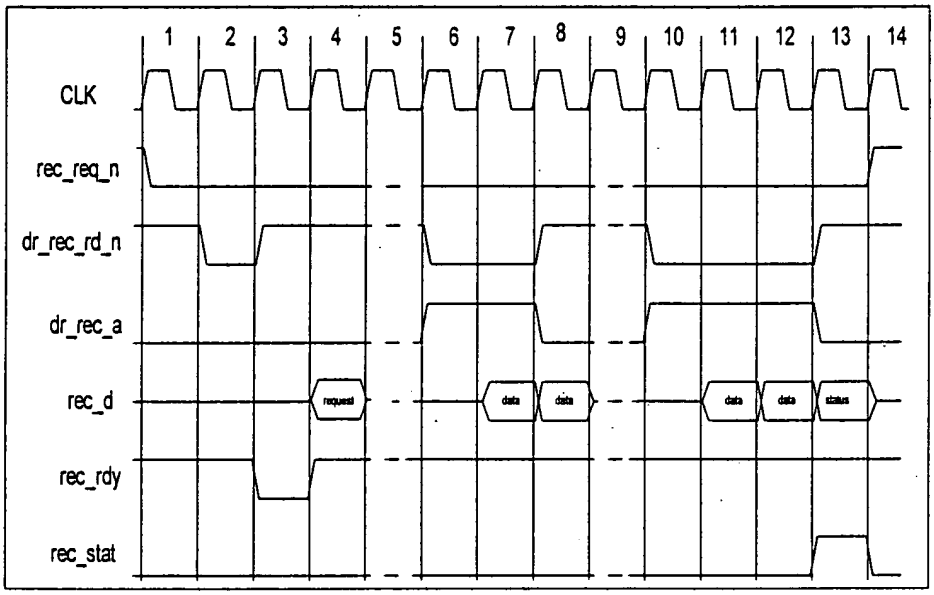


Figure 3
Data Transfer to 2 different Receive Descriptors of 1 channel

In case of a status the word has the following format:

Table 5
Status word

Bit	31	30	29	28	27	26	25	24	23...16	15:8	7:0
Function	FE	MFL	RFOD	CRC	ILEN	RAB	rsvd	rsvd	Byte 2	Byte 1	Byte 0

The BE Bits in the request register of the RB indicate the valid bytes in this statusword. RAB (Receive Abort), ILEN (Invalid Length), CRC (CRC error), RFOD (Receive Frame Overflow), MFL (Maximum Frame Length) and FE (Frame End) are status bits generated by the protocol handler and are mapped in the interrupt vector (refer to chapter chapter 2.4). In case of external or internal problems, indicated by the bits RAB, ILEN, CRC, RFOD, MFL, the FE bit (within the status word) will not be set (exception Receive OFF indication FE and RAB set). When the DMUR updates the current receive descriptor with the error status it then sets the FE bit. New data for this channel, indicated by a new request, are related to a new frame.

3.2.4 DMUI Interface

The DMUR transfers the interrupt vectors to the DMUI (Interrupt controller) which will arbitrate the FPI initiator bus and transfer the vector in the corresponding interrupt queue in the shared memory. The interface between DMUR and DMUI is a bus shared by all the blocks generating interrupts. Therefore the transfer starts with an arbitration cycle. Once the bus is granted to DMUR, it deactivates its request line and writes the vector to the interrupt vector data bus in the next cycle (because no address cycle is needed). The interrupt controller will not insert wait states. Since more than one block is working on the interrupt bus, it might take some cycles until the request is granted. The minimum latency will be one clock.

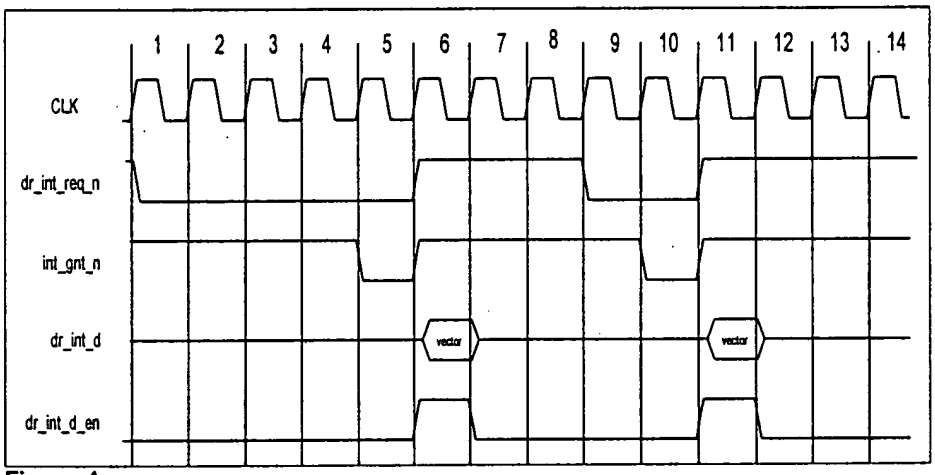


Figure 4
Interrupt Vector Transfer to the DMUI

3.3 Macro Functional Test

3.4 Macro Production Test

4 Register Description

4.1 Register Overview

The DMUR macro will be used in a system. Within the system it will happen that certain configuration information will be used by more than one macro. System programming would be very ineffective if these configuration information will be stored at different locations in different macros. That's why a broadcast programming will be introduced. Channel configuration information will be stored in a set of registers, selected by the signal tfpi_vc_sel_n. Another set of registers, selected by the signal tfpi_vg_sel_n, contains global information for all macros and testmode information. Not all channel and general information are relevant for the DMUR macro, every macro implements only the registers which contain necessary information.

All registers will be accessed via the target FPI bus, inside the macro the SMIF-BPI interface will control this accesses.

Essential for all slave register accesses are fast write accesses (i.e. no PCI wait states) and that a mechanism is provided to ensure that accesses are completed internally before the next command is given (i.e. no erroneous overwrite of data).

Read accesses are non critical, but a debug read back of all registers (virtual and global) must be ensured;

Test mode accesses are non critical and can deliver deliberate results; i.e. read back RAM contents

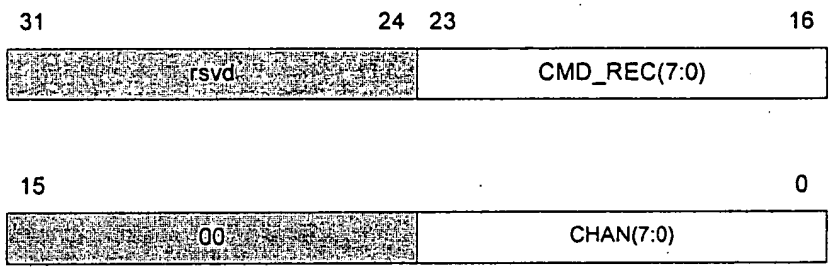
The following chapter describes a subset of registers, which are necessary for programming the DMUR macro. The reset values are the driven values from the DMUR, in the system environment they might be different (non implemented bits by the DMUR might vary).

4.2 Detailed Register Description of the Virtual Channel Specification (selected by tfpi_vc_sel_n)

4.2.1 Virtuell Channel Specification Command Register (CSPEC_CMD)

Access	: read/write
Address	: 00_H

Reset Value : 00000000_H



Note: The reserved bits are implemented by other macros within the system

The following commands will be supported:

Receive Init

This command will cause the initialization of the particular channel CHAN(7:0). The DMUR will store the First Receive Descriptor Address (Register CSPEC_FRDA) in its on-chip memory. When this task is completed a Command Completed Interrupt CMDC vector is transferred to DMUI. Afterwards the DMUR will wait for the first request for this channel from the Receive Buffer. Based on this request the DMUR will fetch the entire receive descriptor pointed by the CSPEC_FRDA register.

Receive Abort

If a receive abort is detected the current receive descriptor will be suspended. The DMUR stores the FRDA in its on chip RAM and generates a CMDC Interrupt.

In transparent mode, an Receive Abort Interrupt will be generated (RAB and HI if necessary), the C bit will be set in the old receive descriptor and the RAB bit is set to one in the status. Then the DMUR jumps immediately to the Receive Descriptor pointed by FRDA triggered by the next request from the receive buffer for this channel. The data from this request will be sent to the data section of this new receive descriptor.

For frame based protocolls (HDLC,PPP) two scenarios are possible. With the next request for this channel, the DMUR recognizes the Abort Command. When the DMUR has finished the previous transmission for this channel with an end of frame (receive descriptor already updated, interrupt generated), new data for this channel are related to the new receive descriptor pointed by FRDA. If the last transferred data for this channel did not contain an frame end, the rest of the received frame, which was only transferred partially, will be discarded. The C and FE bit will be set in this receive descriptor, the status contains the activated RAB bit. The same information will be sent in an Interrupt vector (RAB, FE and HI - if necessary, set). With the next frame the DMUR will branch to the next receive descriptor pointed by the FRDA.

If the data section of the last receive descriptor was filled completely (does not contain a end of frame), the complete bit will be set in the status of this descriptor. The CPU issues a Receive Abort Command before the next request for this channel will appear from the receive buffer. In this case, the DMUR retrieves the receive descriptor pointed by FRDA. This descriptor will be used in order to inform the CPU about the aborted frame with BNO=0 (no data will be written to the data section), the C Bit, the RAB in the status and the FE (only for HLDC). Additionally the corresponding interrupt will be generated. The datsection of the next receive descriptor (pointed by next receive descriptor pointer) will be used in order to store the data of the current request (transparent mode) or the data of the next new frame.

This command also can be used in order to release a channel from the HOLD state. The HOLD descriptor will not be touched anymore since the update already took place, the HOLD bit will not be repolled. No interrupt will ge generated (besides the CMDC), in TMA mode the DMUR jumps immediately to the new receive descriptor pointed by FRDA, in frame based protocolls data of a new frame are related to the receive descriptor pointed by FRDA.

Receive Off

This command will not be durectly interpreted by the DMUR. As long as the DMUR does not see the Receive OFF status word (FE+ Abort bit set; if end of frame and receive off occur at the same time in the protocoll machine, the receive off command will be given priority and so only one receive off status word (FE + Abort set) will be forwarded to the DMUR), it works normally.

If this status word is seen by the DMUR and a frame has not been transferred completely and the current receive descriptor has not been updated yet, it will generate a Receive Abort Interrupt (FE, RAB and HI, if necessary, set). The current receive descriptor will be released setting the C, FE and write the abort status. Afterwards a Command Complete interrupt will be generated.

If the Receive Off statusword is seen by the DMUR and a frame has not been transferred completely and the current receive descriptor already has been updated, the DMUR retrieves the next receive descriptor, writes BNO with 0 and sets FE, C and RAB in the status. Additonally the corresponding Receive Abort Interrupt (FE, RAB and HI, if necessary) will be generated, afterwards the CMDC Interrupt.

If this receive off statusword (FE and Abort set) is seen by the DMUR and the frame was transferred completely, just the CMDC interrupt will be generated.

The channel is turned off. The next command for this channel must be an Receive Init. The DMUR does not expect any data for the channel after the receive off status word was read by the macro until the next initialization.

The Receive Off command also can be used to release a channel, which is on HOLD. No matter if a frame was transferred completely or not, just a CMDC Interrupt will be generated.

Receive Debug

All the registers (besides CSPEC_MODE_REC, since PMD value will be implemented and used by various macros - other macro drives this value during read) of the virtual channel specification will be read with this command.

Receive Hold Reset

Always when the CPU removes the HOLD bit in the receive descriptor chain of the channel additionally it has to write the Receive Hold Reset Command. Internal action refer to description of the polling mechanism (chapter 2.2).

Issuing the Receive Hold Reset Command bit will be accepted immediately, but no acknowledgement (CMDC INTR) will be performed. After executing one of the other commands (Init, Abort, Off) an Command Completed Interrupt (CMDC) will be generated. The CPU may not write another command for this channel into the command register before the CMDC INTR occured. The commands are coded in the following way:

Command Table Receive:

23	22	21	20	19	18	17	16	
Rsvd	Rsvd	Rsvd	Debug	HOLD RESET	ABORT	OFF	INIT	function
0	0	0	0	0	0	0	1	receive init
0	0	0	0	0	0	1	0	receive off
0	0	0	0	0	1	0	0	receive abort
0	0	0	0	1	0	0	0	receive hold reset
0	0	0	1	0	0	0	0	receive debug
0	0	0	0	0	0	0	0	receive nop

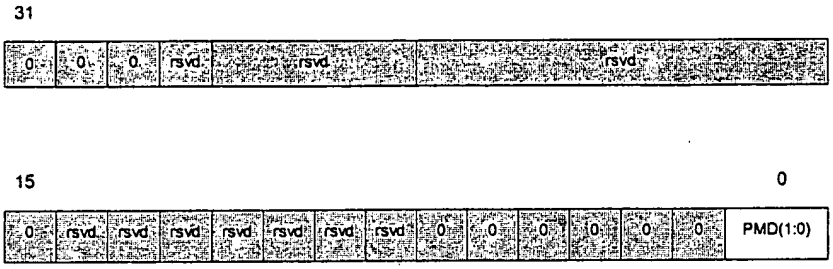
With the writing of Virtual Channel Specification Command Register the contents of all registers belonging to Virtual Channel Specification Broadcast programming is valid.

4.2.2 Virtuell Channel Specification Mode Receive Register (CSPEC_MODE_REC)

Access : write

Address : 04_H

Reset Value : 00000000_H



Note: The reserved bits are implemented by other macros within the system

PMD(1:0): Protocoll Machine Mode

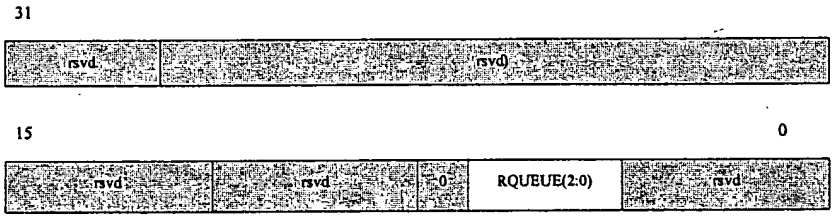
For each channel, an information is needed whether the protocol is frame based (HDLC, PPP, Ethernet,...) or transparent. In the transparent mode no status byte containing FE is received from RB.

4.2.3 Virtual Channel Specification Buffer (CSPEC_BUFFER)

Access : read/write

Address : 20_H

Reset Value : 00000000_H



RQUEUE: The generated receive interrupts will be sent to this interrupt queue

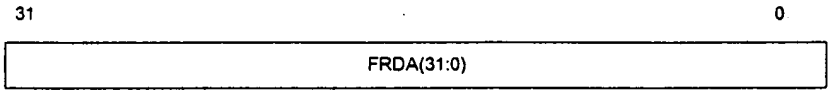
Note: The reserved bits are implemented by other macros within the system.

4.2.4 Virtual Channel Specification FRDA (CSPEC_FRDA)

Access : read/write

Address : 24_H

Reset Value : 00000000_H



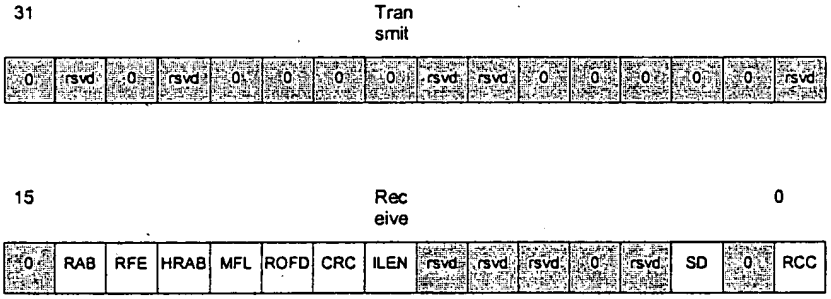
FRDA(31:0): First Receive Descriptor Address (dword aligned)

4.2.5 (Virtual) Channel Specification Interrupt Mask

Access : read/write

Address : $2C_H$

Reset Value : 00000000_H



Interrupt Generation Mask: These bits correspond to the respective channel and command interrupts, if set to '1', the corresponding interrupt will not be generated by the device

Note: The reserved bits are implemented by other macros within the system

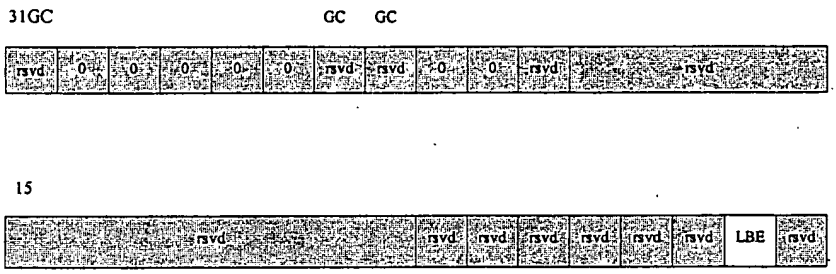
4.3 Detailed Register Description of the Global registers (selected by tfpi_vg_sel_n)

4.3.1 Configuration Register 1 (CONF1)

Access : read/write

Offset Address : 40_H

Reset Value : 00000000_H



LBE: Little Endian/Big Endian Byte Swap; Big Endian: LBE = 1

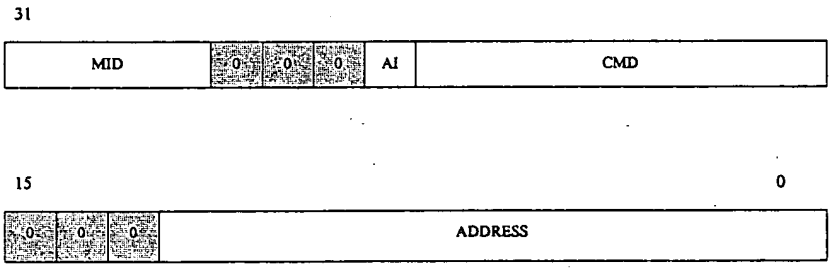
Note: The reserved bits are implemented by other macros within the system

4.3.2 Test Command Register (TAC)

Access : write/read

Offset Address : 58_H

Reset Value : 00000000_H



MID: Macro ID Code (0101 for DMUR)

AI: Auto Increment Function

Address: internal address

CMD: Command (select of ram and register)

CMD:

23	22	21	20	19	18	17	16	function
0	0	0	0	0	0	0	1	RAM1_NO
0	0	0	0	0	0	1	0	RAM1_NDPTR
0	0	0	0	0	0	1	1	RAM1_RDPTR
0	0	0	0	0	1	0	0	RAM1_CDPTR
0	0	0	0	0	1	0	1	RAM2
0	0	0	0	0	1	1	0	R1_REG
0	0	0	0	0	1	1	1	R2_REG
0	0	0	0	1	0	0	0	ADDR_REG
0	0	0	0	1	0	0	1	IV_REG
0	0	0	0	1	0	1	0	NO_REG
0	0	0	0	1	0	1	1	BNO_REG
0	0	0	0	1	1	0	0	STRA_REG
0	0	0	0	1	1	0	1	TNOBL_REG
0	0	0	0	1	1	1	0	FPID_REG
0	0	0	0	1	1	1	1	FPIA_REG

Macro ID Code: Bit 31:28 = '0101' for DMUR

The Address in the Command Register is only for RAMx accesses (command 1 - 5) valid, address width is 8 bit. Autoincrement will only be supported for RAM Accesses.

General

- the test access provides read/write access of important internal rams and registers
- test registers are virtuals global registers (SEL signal: pb_vg_tfpi_sel_n / implemented as a daisy chain).
- the single macros are selected by the MID code of the test command register
- the test access provides read/write access of important internal rams and registers
- CMD specifies/selects one of the macro rams/registers
- the address field is used to access a ram address

- AI: autoincrement : address given in the address field is incremented automatically for each access
- All macros which are not selected by MID drive data output "00000000" and <macro>_TPFI_RDY = '1'. Driving "00000000" would mean not disable the enable line for data out, but to set the output data to "00000000".

Test write access:

1. Write TAC
2. Write TAD

Test read access:

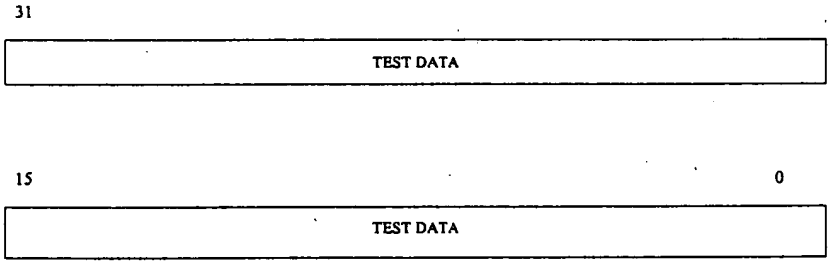
1. Write TAC
2. Read TAD

Typically the macro selected via MID delays the RDY signal until the selected ram/ register has been read and the data can be provided at the TFPI interface. No prefetch of testdata is required.

Note: CMD/Address have to be defined for each macro; there is no read/write selection in the CMD field; rd/wr's are handled with the TFPI read & write signals

4.3.3 Test Data Register (TD)

Access	: read/write
Address	: $5C_H$
Reset Value	: 00000000_H



TEST DATA: ram/register test data (read or write)